Tutorials Exercises Services Sign Up Log in ERY EXCEL XML DJANGO NUMPY

PANDAS NODEJS DSA TYPESCRIPT A

W3Schools Certification Course CHECK IT OUT!

R Vectors

❮ Previous Next ❯

Vectors

A vector is simply a list of items that are of the same type.

To combine the list of items to a vector, use the `c()` function and separate the items by a comma.

In the example below, we create a vector variable called **fruits**, that combine strings:

Example

```
# Vector of strings
fruits <- c("banana", "apple", "orange")

# Print fruits
fruits
```

Try it Yourself »

In this example, we create a vector that combines numerical values:

Example

```
# Vector of numerical values
numbers <- c(1, 2, 3)
```



```
# Print numbers
numbers
```

Try it Yourself »

To create a vector with numerical values in a sequence, use the `:` operator:

Example

```
# Vector with numerical values in a sequence
numbers <- 1:10

numbers
```

Try it Yourself »

You can also create numerical values with decimals in a sequence, but note that if the last element does not belong to the sequence, it is not used:

Example

```
# Vector with numerical decimals in a sequence
numbers1 <- 1.5:6.5
numbers1

# Vector with numerical decimals in a sequence where the last element is
not used
numbers2 <- 1.5:6.3
numbers2
```

Result:

```
[1] 1.5 2.5 3.5 4.5 5.5 6.5
[1] 1.5 2.5 3.5 4.5 5.5
```

 In the example below, we create a vector of logical values:

Example

```
# Vector of logical values
log_values <- c(TRUE, FALSE, TRUE, FALSE)

log_values
```

Try it Yourself »

Vector Length

To find out how many items a vector has, use the `length()` function:

Example

```
fruits <- c("banana", "apple", "orange")

length(fruits)
```

Try it Yourself »

Sort a Vector

To sort items in a vector alphabetically or numerically, use the `sort()` function:

Tutorials □ Exercises □ Services □ □ Sign Up Log in

Example

ERY EXCEL XML DJANGO NUMPY PANDAS NODEJS DSA TYPESCRIPT A

```
fruits <- c("banana", "apple", "orange", "mango", "lemon")
numbers <- c(13, 3, 5, 7, 20, 2)

sort(fruits) # Sort a string
sort(numbers) # Sort numbers
```

Try it Yourself »

Access Vectors

You can access the vector items by referring to its index number inside brackets `[]` . The first item has index 1, the second item has index 2, and so on:

Example

```
fruits <- c("banana", "apple", "orange")

# Access the first item (banana)
fruits[1]
```

Try it Yourself »

You can also access multiple elements by referring to different index positions with the `c()` function:

Example

```
fruits <- c("banana", "apple", "orange", "mango", "lemon")

# Access the first and third item (banana and orange)
fruits[c(1, 3)]
```

 You can also use negative index numbers to access all items except the ones specified:

Example

```
fruits <- c("banana", "apple", "orange", "mango", "lemon")

# Access all items except for the first item
fruits[c(-1)]
```

Try it Yourself »

Change an Item

To change the value of a specific item, refer to the index number:

Example

```
fruits <- c("banana", "apple", "orange", "mango", "lemon")

# Change "banana" to "pear"
fruits[1] <- "pear"

# Print fruits
fruits
```

Try it Yourself »

Repeat Vectors

To repeat vectors, use the `rep()` function:

Tutorials □ Exercises □ Services □ □ Sign Up Log inExample

ERY☎ EXCEL XML DJANGO NUMPY PANDAS NODEJS DSA TYPESCRIPT A Repeat each value:

```
repeat_each <- rep(c(1,2,3), each = 3)

repeat_each
```

Try it Yourself »

Example

Repeat the sequence of the vector:

```
repeat_times <- rep(c(1,2,3), times = 3)

repeat_times
```

Try it Yourself »

Example

Repeat each value independently:

```
repeat_indepent <- rep(c(1,2,3), times = c(5,2,1))

repeat_indepent
```

Try it Yourself »

Generating Sequenced Vectors

One of the examples on top, showed you how to create a vector with numerical values in a sequence with the `:` operator:

Tutorials Exercises Services Sign Up Log in
ERY EXCEL XML DJANGO NUMPY PANDAS NODEJS DSA TYPESCRIPT A

Example

```
numbers <- 1:10

numbers
```

Try it Yourself »

To make bigger or smaller steps in a sequence, use the `seq()` function:

The Only Trading E
Book You Need

Example

```
numbers <- seq(from = 0, to = 100, by = 20)

numbers
```

Download Free

Try it Yourself »

Note: The `seq()` function has three parameters: `from` is where the sequence starts, `to` is where the sequence stops, and `by` is the interval of the sequence. *T&Cs Apply

 Track your progress - it's free! Sign Up Log in